Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-14
DOI:10.22034/cmde.2026.70547.3514

An efficient computational scheme combining legendre polynomials and deep neural networks for nonlinear integral equations

Zeinab Hajimohammadi[1,2,*] **and Koroush Parand**[2,3]

[1]Interdisciplinary Quranic Studies Research Institute, Shahid Beheshti University, Tehran, Iran.

[2]Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran.

[3]Institute for Cognitive and Brain Sciences, Shahid Beheshti University, Tehran, Iran.

Abstract

Various phenomena in biology, physics, and engineering are modeled by differential equations. These differential equations, including partial differential equations and ordinary differential equations, can be converted and represented as integral equations. In particular, Volterra–Fredholm–Hammerstein integral equations are a main class of such integral equations, and researchers are interested in investigating and solving these equations. In this paper, we propose Legendre Deep Neural Network (LDNN) for solving nonlinear Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs). LDNN utilizes Legendre orthogonal polynomials as activation functions in the deep structure. We present how LDNN can be used to solve nonlinear V-F-H-IEs. We show that using the Gaussian quadrature collocation method in combination with LDNN results in a novel numerical solution for nonlinear V-F-H-IEs. Several examples are given to verify the performance and accuracy of LDNN.

Keywords. Deep Neural Network, Legendre polynomials, Collocation method, Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs).
2010 Mathematics Subject Classification. 65R20, 45B05, 45D05, 68W25, 68T05, 68T30.

Uncorrected Proof

1. Introduction

Various phenomena in biology, physics, finance, neuroscience, and engineering are modeled by differential equations [3, 13, 16, 17, 26]. Integral models and their applications have been studied in many fields of science, mathematics, engineering, and technology, and research in this field has expanded greatly [2, 4, 8, 9, 27, 41, 48, 65].
Various numerical methods have been applied for solving differential and integral equations. The Homotopy analysis method (HAM) [37] and the variational iteration method (VIM) [24] are known as analytical/semi-analytical methods. Usually, the Nystrom method [34], spectral methods [11, 43, 48, 51, 52], Runge–Kutta methods [20], finite difference methods (FDM) [61], and finite element methods (FEM) [28] are considered as popular numerical methods. When the complexity of the model does not allow us to obtain the solution explicitly, numerical methods are a proper selection for finding the approximate solution for the model. Deep neural networks are an important part of the machine learning family and are applied in various areas including engineering, speech processing, computer vision, natural language processing, image processing, and related areas [15, 18, 31, 35, 36, 53]. In addition, function approximation is a significant branch of scientific computing and has been considered in several studies [23, 47, 62]. Solving differential equations is another important branch of scientific computing, and deep learning has shown success in this area [10, 32, 33, 55, 67]. In recent years, several researchers have studied solving differential equations via deep learning or neural networks. Differential equations include ordinary differential equations, partial differential equations, and integral equations [32, 40, 45, 60].

Received: 05 December 2025; Accepted: 01 November 2026.
∗ Corresponding author. Email: Z_Hajimohammadi@sbu.ac.ir.

Recently, some of the machine learning methods have been applied to solving differential equations. Chakraverty and Mall [12] introduced orthogonal neural networks, which used orthogonal polynomials in the structure of the network. Raja *et al.*[56] applied the meta-heuristic optimization algorithm to the neural network to obtain the solution of differential equations. Moreover, other methods of machine learning, such as support vector machine [63], are used to approximate the solution of the models. Least squares support vector machines are considered in these studies [21, 44]. Baker *et al.*[5] selected deep neural networks for solving the differential equations. Pang *et al.*[50] introduced a new network to find the solution to the different equations. Han *et al.*[22] solved high-dimensional problems via deep networks. Also, Long *et al.*[39] and Raissi *et al.*[55] introduced a group of equations that were solved by deep learning. Furthermore, He *et al.*[25] and Molina *et al.*[49] investigated the effect of the activation function on networks.
LSTM-RNNs have been used to approximate nonlinear integral operators in integro-differential equations, enabling more efficient solutions and reduced computational complexity. Bassi *et al.*[6] used this approach to solve the Dyson equation in quantum many-body systems.
This study [46] presents a Double Deep Q-Network (DDQN)-based framework for discovering interpretable symbolic solutions to nonlinear integral and integro-differential equations, integrating context-free grammars and physics-informed neural networks for structured expression generation and coefficient optimization. The results show improved stability, convergence, and interpretability over standard DQN, particularly in complex engineering applications.
An artificial neural network (ANN) framework based on shifted Gegenbauer polynomials has been proposed for solving various types of integral equations, combining classical optimization and collocation training methods [54]. This approach demonstrates strong accuracy and convergence, outperforming several traditional numerical techniques.
This paper [1] introduces a Chebyshev polynomial-based least-squares support vector regression (LSSVR) method using operational matrices to efficiently solve fractional integro-differential equations. The approach reformulates the problem into a matrix-based framework, achieving faster computation with high accuracy and exponential convergence, and proves effective for nonlinear and multi-dimensional fractional problems.
Furthermore, Mall *et al.*[42] considered that connectionist learning models, particularly Functional Link Artificial Neural Networks (FLANN) and Laguerre Neural Networks (LgNN), can be effectively applied to solve differential and integral equations, including systems of Fredholm integral equations. These ANN-based methods improve performance in solving various nonlinear problems, such as boundary value problems, Lane–Emden equations, and Duffing oscillator models.
Advanced numerical methods, including spectral collocation, kernel approximation, and hybrid or AI-based techniques, have been developed to solve nonlinear Fredholm integral equations arising in engineering applications [30]. These approaches enhance stability and accuracy through strategies such as adaptive mesh refinement, Gaussian quadrature, and machine learning, demonstrating effectiveness across areas like structural mechanics and biomedical engineering.
Dana Mazraeh *et al.*[14] introduced FIE-500k, a large symbolic dataset for second-kind Fredholm integral equations, generated using context-free grammars to ensure valid and diverse mathematical expressions. The dataset supports training and evaluation of language models for symbolic modeling and solving of integral equations.
Ren *et al.*[58] introduced the Enhanced Physics-Informed Neural Network (EPINN). EPINN is designed to efficiently solve high-dimensional and nonlinear integral equations by combining operator decomposition, Sobolev-space projection, and boundary-aware multi-objective training. It achieves significantly higher accuracy and computational efficiency than traditional numerical methods and existing deep learning solvers across various Volterra and Fredholm benchmark problems.

In this paper, we consider nonlinear Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs) and obtain their solutions via a deep neural network. We present a new numerical approach of machine learning, which is a combination of a deep neural network and the Legendre collocation method. This approach is useful for solving differential equations and is applied here to solve nonlinear V-F-H-IEs. We incorporate the Legendre collocation method into the network to improve the numerical computations and enhance its performance.

2. Legendre deep neural network (LDNN)

The main purpose of introducing LDNN is to apply it to solving differential models. This aims to expand the utilization of deep learning networks in the field of scientific computing, especially the solution of differential equations.

Uncorrected Proof

Moreover, this network has the advantages of solving equations by deep learning as well as numerical methods, such as the collocation method, used to achieve better solutions to the equations. LDNN presents a combination of a deep neural network and the Legendre collocation method. The proposed network consists of two subnetworks that are connected consecutively. The first network is a feed-forward neural network with an orthogonal Legendre layer. The second network includes operation nodes that create the desired computational model. In recent decades, numerical methods, especially the collocation method, have been popular methods for solving differential equations. In the collocation method, first, an approximation of the solution is expanded by using the sum of the basic functions. The basis functions consist of orthogonal polynomials such as Legendre polynomials. This approximation is then substituted into the differential equation. By considering the appropriate set of candidate points, an attempt is made to obtain the unknown coefficients of the basic functions so that the solution satisfies the equation in a set of candidate points. The first network is applied to approximate the solution. This approximation can be regarded as a scattered data interpolation problem. The second network is used to construct the desired equation so that the solution satisfies it. The structure of LDNN is described in detail in the following.
Consider that the first network has an $\mathcal{M}$-layer which is defined as follows:

$$\begin{aligned}
&\mathcal{H}_0 = x, \quad x \in \mathbb{R}^d,\\
&\mathcal{H}_1 = L(W^{(1)}\mathcal{H}_0 + b^{(1)}),\\
&\mathcal{H}_i = f(W^{(i)}\mathcal{H}_{i-1} + b^{(i)}), \quad 2 \le i \le \mathcal{M}-1,\\
&\mathcal{H}_{\mathcal{M}} = W^{(\mathcal{M})}\mathcal{H}_{\mathcal{M}-1} + b^{(\mathcal{M})}.
\end{aligned}$$

where $\mathcal{H}_0$ is the input layer with d dimensions. $\mathcal{H}_i$, $1 \le i \le \mathcal{M}-1$ are hidden layers, $L = [L_0, L_1, ...L_n]^T$ where L_i are i-th degrees of Legendre orthogonal polynomials, $\mathcal{H}_1$ is an orthogonal layer, f is the hyperbolic tangent activation function or other commonly used activation functions. $W^{(i)}$, $i = 1, \cdots, \mathcal{M}$ are the weight parameters and $b^{(i)}$, $1 \le i \le \mathcal{M}$ are the bias parameters. $\mathcal{H}_{\mathcal{M}}$ is the output layer. Notably, the second network is applied to obtain the desired differential model. This aim is possible by using operation nodes including integrals, derivatives, etc. These nodes are applied to the output of the first network. Moreover, automatic differentiation (AD) [7] and Legendre Gaussian integration [59] have been used in network computing to obtain more accurate and faster calculations. How to train the network and set the parameters are also important points. Supervised learning method is used to train the network. The cost function for setting parameters is defined as follows:

$$\text{CostFun} = \min(y_t - y_p) + \min(R_m). \tag{2.1}$$

where y_t is an exact value of the model, and y_p is a predicted value of the LDNN. The definition of R_m is explained in section 3. The minimization of CostFun is obtained by performing the Adam algorithm [29] and the L-BFGS method [38] on the mean squared errors of the training dataset.

Uncorrected Proof

2.1. **Legendre polynomials.** Legendre polynomials [59] are a main series of orthogonal polynomials, which are denoted by $L_n(\eta)$, and are defined as:

$$L_n(\eta) = \frac{1}{2^n}\sum_{\ell=0}^{[\frac{n}{2}]}(-1)^\ell \frac{(2n-2\ell)!}{2^n \ell!(n-\ell)!(n-2\ell)!}\eta^{n-2\ell}. \tag{2.2}$$

Legendre polynomials are defined in the $[-1, 1]$ domain and have the recurrence formula in the following form:

$$\begin{aligned}
&(n+1)L_{n+1}(\eta) = (2n+1)\eta L_n(\eta) - nL_{n-1}(\eta), \quad n \ge 1,\\
&L_0(\eta) = 1, \qquad L_1(\eta) = \eta.
\end{aligned} \tag{2.3}$$

Orthogonality relation for these polynomials is as follows:

$$\int_{-1}^{1} L_n(\eta)L_m(\eta)\mathrm{d}\eta = \gamma\delta_{n,m}, \tag{2.4}$$

where $\delta_{n,m}$ is a delta Kronecker function and $\gamma = \frac{2}{2n+1}$.
The weight function of them is $\mathcal{W}(\eta) = 1$. The following useful properties of Legendre polynomials are defined:

$$L_n(-\eta) = (-1)^n L_n(\eta), \tag{2.5}$$
$$|L_n(\eta)| \leq 1, \quad \forall \eta \in [-1,1],\ n \geq 0, \tag{2.6}$$
$$L_n(\pm 1) = (\pm 1)^n, \tag{2.7}$$
$$(2n+1)L_n(\eta) = L'_{n+1}(\eta) - L'_{n-1}(\eta), \quad n \geq 1. \tag{2.8}$$

3. Nonlinear Volterra–Fredholm–Hammerstein integral equations and LDNN

The general form of nonlinear Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs) is as follows:

$$y(x) = g(x) + \xi_1 \int_0^x K_1(x,s)\varphi_1(s,y(s))\mathrm{d}s + \xi_2 \int_0^1 K_2(x,s)\varphi_2(s,y(s))\mathrm{d}s, \quad x \in [0,1]. \tag{3.1}$$

where ξ_1, ξ_2 are fixed, $g(x)$, $K_1(x,s)$ and $K_2(x,s)$ are given functions and $\varphi_1(s,y(s))$, $\varphi_2(s,y(s))$ are nonlinear functions. The aim is to find the proper $y(x)$. To use the LDNN, reformulate Eq. (3.1) in the following form:

$$R_m = -y(x) + g(x) + \xi_1 \int_0^x K_1(x,s)\varphi_1(s,y(s))\mathrm{d}s + \xi_2 \int_0^1 K_2(x,s)\varphi_2(s,y(s))\mathrm{d}s, \quad x \in [0,1]. \tag{3.2}$$

$y(x)$ is approximated by the first network of the LDNN.

$$y(x) \approx \mathcal{H}_{\mathcal{M}}. \tag{3.3}$$

Furthermore, we applied the Legendre–Gauss integration formula [59]:

$$\int_{-1}^1 h(X)\mathrm{d}X = \sum_{j=0}^N \omega_j h(X_j), \tag{3.4}$$

where $\{X_j\}_{j=0}^N$ are the roots of L_{n+1} and $\{\omega_j\}_{j=0}^N = \frac{2}{(1-X_j^2)(L'_{n+1}(X_j))^2}$. Here, we should transfer the $[0,x]$ and $[0,1]$ domains into the $[-1,1]$ domain. It is possible by using the following transformation:

$$t_1 = \frac{2}{x}s - 1, \quad t_2 = 2s - 1. \tag{3.5}$$

Consider

$$Z_1(x,s) = K_1(x,s)\varphi_1(s,y(s)),$$
$$Z_2(x,s) = K_2(x,s)\varphi_2(s,y(s)).$$

we have

$$R_m = -y(x) + g(x) + \xi_1 \frac{x}{2}\int_{-1}^1 Z_1(x, \frac{x}{2}(t_1+1))\mathrm{d}t_1 + \frac{\xi_2}{2}\int_{-1}^1 Z_2(x, \frac{x}{2}(t_2+1))\mathrm{d}t_2. \tag{3.6}$$

By using the Legendre–Gauss integration formula, the following form is obtained:

$$R_m = -y(x) + g(x) + \xi_1 \frac{x}{2}\sum_{j=0}^{N_1} \omega_{1j} Z_1(x, \frac{x}{2}(t_{1j}+1)) + \frac{\xi_2}{2}\sum_{j=0}^{N_2} \omega_{2j} Z_2(x, \frac{x}{2}(t_{2j}+1)). \tag{3.7}$$

The second network of LDNN and its nodes make R_m. The architecture of LDNN for solving nonlinear V-F-H-IEs is represented in Figure 1.

4. Numerical results

To present the accuracy and performance of the LDNN for solving nonlinear V-F-H-IEs and justify the efficiency of the proposed method, several examples are given. The convergence behavior of the LDNN is reported by using the following parameters:
The exact value y_t, the predicted value y_p, and the absolute error (Error) in some points of the test data are reported in various tables. The number of the train data m_1, the number of Legendre quadrature points (N_1, N_2), the number

Uncorrected Proof

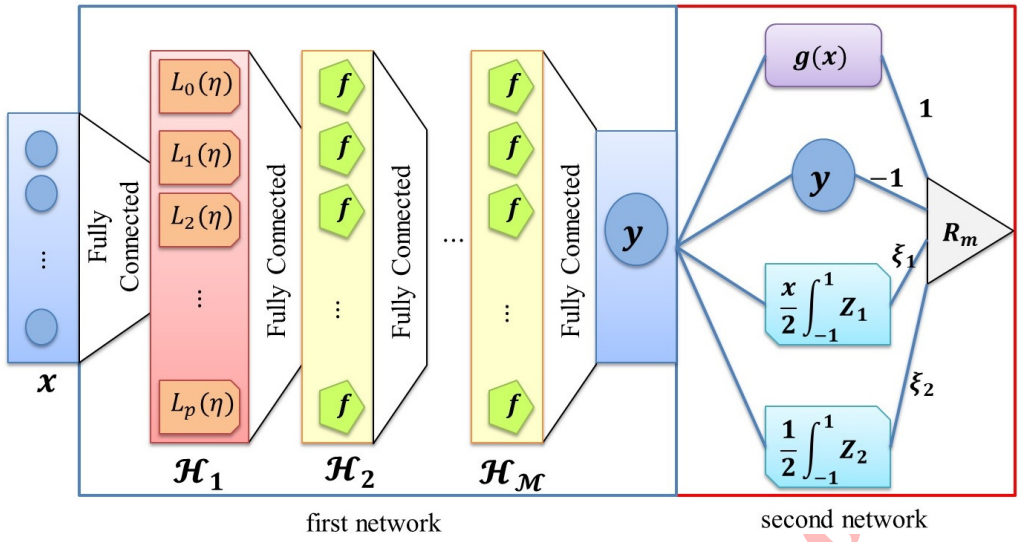


FIGURE 1. The architecture of LDNN for solving nonlinear V-F-H-IEs. The first network approximates the solution of IE $y(x)$. This network has an $\mathcal{M}$-layer feed-forward structure. $\mathcal{H}_1$ is introduced as an orthogonal layer which consists of p neurons with $\{L_i\}_{i=0}^{p}$ (Legendre polynomials) as activation functions. Other layers have f, the hyperbolic tangent as activation functions. The second network uses operation nodes to construct the desired model, and its output is R_m (consider Eq. (3.7)). The outputs of LDNN are $y(x)$ and R_m.

Uncorrected Proof

of the test data m_2, the structure of the network $\mathcal{M}$-layers, L_2^{train} and L_2^{test} are shown in Table 1. L_2^{train} and L_2^{test} are calculated as follows:

$$L_2^{train} = ||y_t - y_p||_2 = [\sum_{j=1}^{m_{tr}} (y_t(x_j) - y_p(x_j))^2]^{\frac{1}{2}},$$
$$L_2^{test} = ||y_t - y_p||_2 = [\sum_{j=1}^{m_{te}} (y_t(x_j) - y_p(x_j))^2]^{\frac{1}{2}}. \tag{4.1}$$

The TensorFlow package with Python version 3.7.0 was used to implement the code of all experiments. The Adam algorithm was terminated after reaching 5000 iterations, and the L-BFGS method was stopped upon convergence. The figures are obtained on the test dataset. For comparison, the results of the LDNN are evaluated against other well-known methods, the Adomian decomposition method (ADM) [64] and a simple Feedforward neural network (FNN) which have supervised training and similar number of parameters and standard loss functions without the IE residual, are considered. The numerical results are reported in the tables.

4.1. **Experiment 1.** Suppose that we have the following model [66]:

$$y(x) = \mathrm{e}^x - \frac{1}{3}\mathrm{e}^{3x} + \frac{1}{3} + \int_0^x y^3(s)\mathrm{d}s, \quad x \in [0,1]. \tag{4.2}$$

It has the exact solution $y(x) = \mathrm{e}^x$. Table 2 represents the exact value, the predicted value, and the absolute error (Error) in several test points in the domain $[0,1]$. 50 points of the shifted Legendre quadrature points are applied for training the LDNN. The number of the training dataset is 500 and the number of the test dataset is 100. Figure 2 shows the illustrated comparison between y_t and y_p.

TABLE 1. The LDNN parameters for all the experiments. The structure of the $\mathcal{M}$-layers is indicated by $[d, NL^{(1)}, NL^{(2)}, \cdots, NL^{(\mathcal{M}-1)}, 1]$. This network has the d dimensions in input layer, $\mathcal{M}-1$ hidden layers with $NL^{(\ell)}$, $2 \leq \ell \leq \mathcal{M}-1$, neurons in each layer and one output which approximates the $y(x)$. All the experiments have 4 hidden layers.

Experiment	$\mathcal{M}$-Layers	m_1	(N_1, N_2)	L_2^{train}	m_2	L_2^{test}
Experiment 1	$[1, 10, 30, 20, 10, 1]$	500	$(50, -)$	3.937867e−09	100	4.015095e−09
Experiment 2	$[1, 10, 30, 20, 10, 1]$	500	$(50, 50)$	7.156029e−09	100	7.537263e−09
Experiment 3	$[1, 10, 30, 20, 10, 1]$	500	$(50, 50)$	1.347132e−09	100	1.659349e−08
Experiment 4	$[1, 10, 30, 20, 10, 1]$	500	$(50, 50)$	9.182442e−09	100	1.107755e−09

TABLE 2. The exact value y_t, the predicted value of the ADM method, the predicted value of the FNN, the predicted value y_p and the absolute error (Error $|y_t - y_p|$) of the LDNN in several test points on the $[0, 1]$ domain for Experiment 4.1.

x	exact value $(y_t = \mathrm{e}^x)$	predicted value by ADM	predicted value by FNN	predicted value (y_p) by LDNN	Error
0.0	1.0	1.0002421	1.0006372	1.000000049	4.90000001e−08
0.2	1.22140276	1.2213538	1.2213246	1.221402765	4.99999997e−09
0.4	1.4918247	1.4919181	1.4919742	1.49182494	2.40000000e−07
0.6	1.8221188	1.8220339	1.8221628	1.82211831	4.90000000e−07
0.8	2.22554093	2.2255747	2.2255346	2.225540981	5.09999998e−08
1.0	2.71828183	2.717803	2.717317	2.71828179	4.00000002e−08

Uncorrected Proof

4.2. **Experiment 2.** Suppose that we have the following model [57]:

$$y(x) = 1 + \sin^2(x) + \int_0^1 K(x, s) y^2(s) \mathrm{d}s, \quad x \in [0, 1]. \tag{4.3}$$

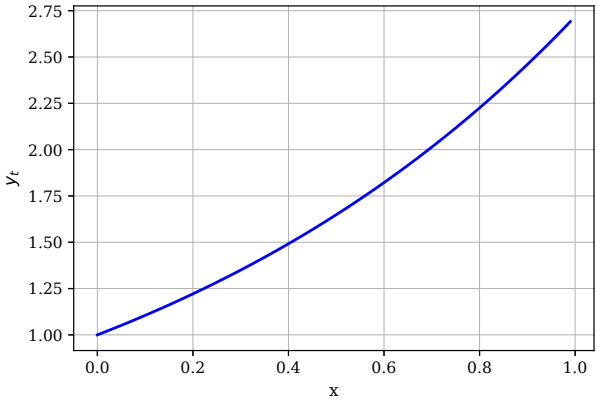
where

$$K(x, s) = \begin{cases} -3\sin(x - s), & 0 \leq s \leq x; \\ 0 & x \leq s \leq 1. \end{cases} \tag{4.4}$$

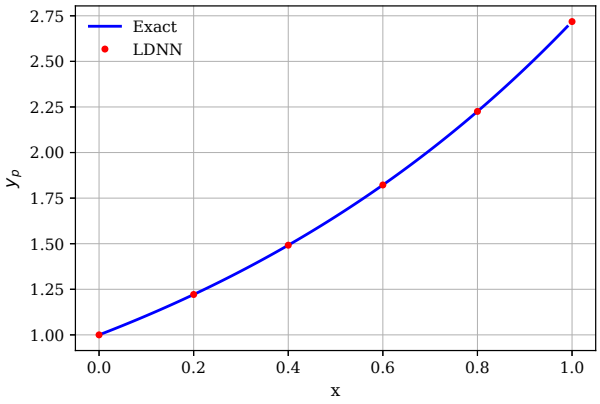
It has the exact solution $y(x) = \cos(x)$. The exact value, the predicted value and the absolute error (Error) in several test points on the $[0, 1]$ domain are reported in Table 3. 50 points of shifted Legendre quadrature points are applied for training LDNN. The number of train data set is 500 and the number of test data set is 100. Figure 3 shows the illustrated comparison between y_t and y_p.

TABLE 3. The exact value y_t, the predicted value of the ADM method, the predicted value of the FNN, the predicted value y_p and the absolute error (Error $|y_t - y_p|$) of the LDNN in several test points on the $[0, 1]$ domain for Experiment 4.2.

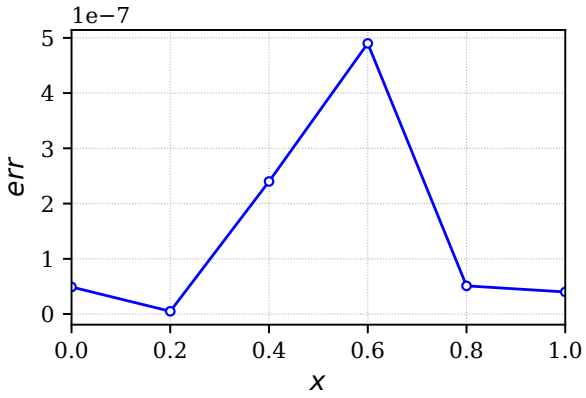
x	exact value $(y_t = \cos(x))$	predicted value by ADM	predicted value by FNN	predicted value (y_p) by LDNN	Error
0.0	1.0	1.0003562	1.0006432	1.000000059	5.90000000e−08
0.2	0.98006658	0.97977763	0.97865423	0.98006683	2.50000000e−07
0.4	0.92106099	0.9210639	0.92105988	0.92106083	1.50000000e−07
0.6	0.82533561	0.82562345	0.82580132	0.82533555	6.00000000e−08
0.8	0.69670671	0.6963889	0.69647832	0.69670670	9.99999994e−09
1.0	0.54030231	0.5411298	0.54212091	0.54030237	6.00000001e−08



(A) Exact solution.



(B) Predicted solution.



(C) Absolute error.

FIGURE 2. Results of Experiment 4.1. Exact solution $y_t(x) = \mathrm{e}^x$, predicted solution $y_p(x)$ by LDNN and absolute error $err = |y_t(x) - y_p(x)|$.

4.3. **Experiment 3.** Suppose that we have the following model [4]:

$$y(x) = g(x) + \int_0^x (x-s)y^2(s)\mathrm{d}s + \int_0^1 (x+s)y(s)\mathrm{d}s, \quad x \in [0,1]. \tag{4.5}$$

where

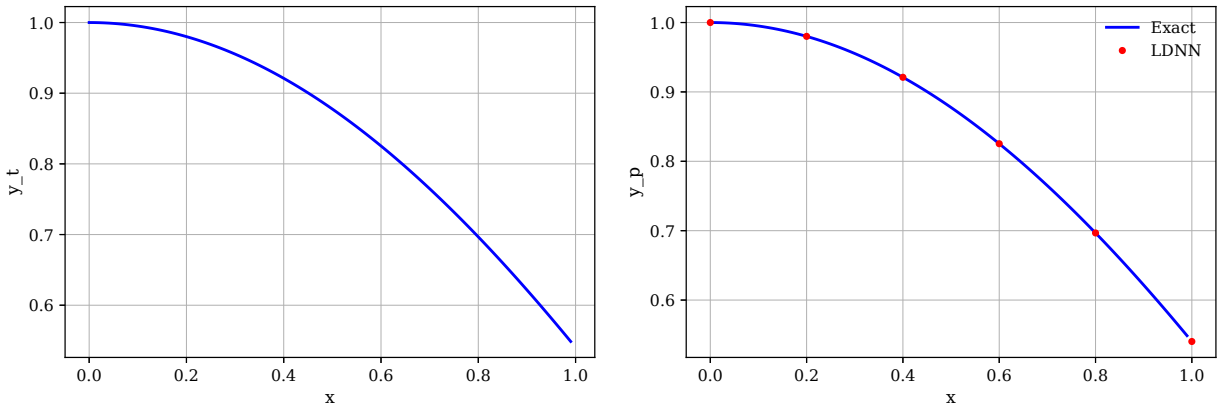
$$g(x) = -\frac{1}{30}x^6 + \frac{1}{3}x^4 - x^2 + \frac{5}{3}x - \frac{5}{4}. \tag{4.6}$$

It has the exact solution $y(x) = x^2 - 2$. Table 4 illustrates the exact value, the predicted value and the absolute error (Error) in several test points on the $[0,1]$ domain. 50 points of the shifted Legendre quadrature points are applied for training LDNN. The number of the training dataset is 500 and the number of the test dataset is 100. Figure 4 represented the comparison between y_t and y_p.

4.4. **Experiment 4.** Suppose that we have the following model [19]:

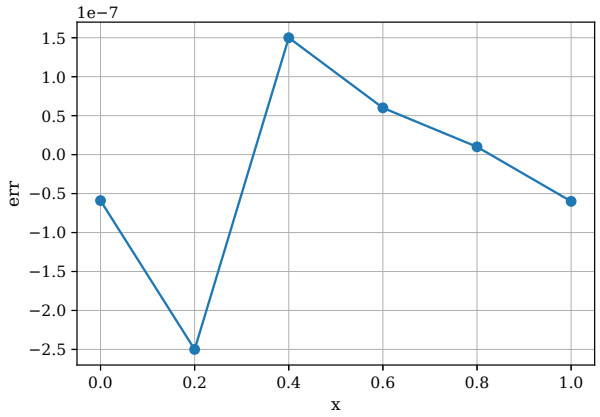
$$y(x) = -\frac{1}{10}x^4 + \frac{5}{6}x^2 + \frac{3}{8} + \int_0^x \frac{1}{2x}y^2(s)\mathrm{d}s, \quad x \in [0,1]. \tag{4.7}$$

Uncorrected Proof



(A) Exact solution.

(B) Predicted solution.



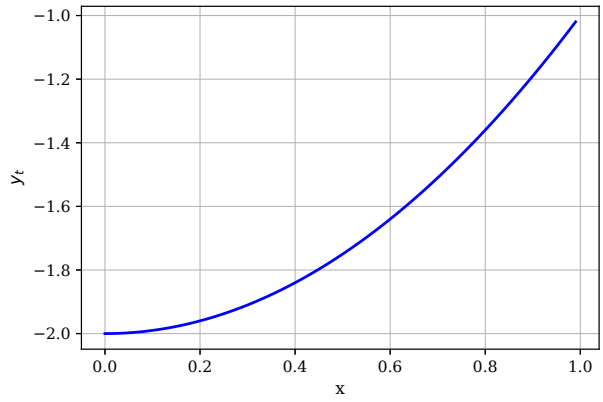
(C) Absolute error.

FIGURE 3. Results of Experiment 4.2. Exact solution $y_t(x) = \cos(x)$, predicted solution $y_p(x)$ by LDNN and absolute error $err = |y_t(x) - y_p(x)|$.

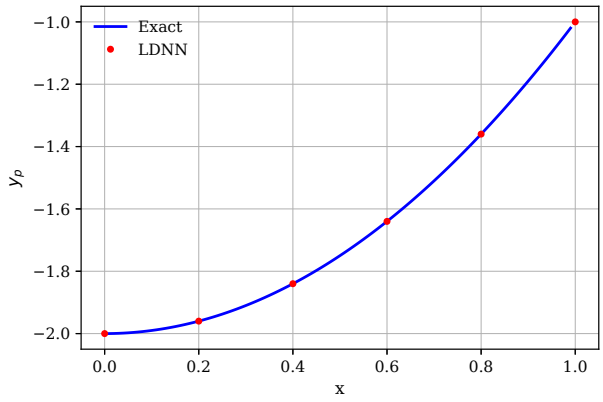
TABLE 4. The exact value y_t, the predicted value of the ADM method, the predicted value of the FNN, the predicted value y_p and the absolute error (Error $|y_t - y_p|$) of the LDNN in several test points on the $[0, 1]$ domain for Experiment 4.3.

x	exact value ($y_t = x^2 - 2$)	predicted value by ADM	predicted value by FNN	predicted value (y_p) by LDNN	Error
0.0	−2.0	−2.0001612	−2.0003422	−2.00000001	9.99999994e−09
0.2	−1.96	−1.960051	−1.9601312	−1.96000049	4.90000000e−07
0.4	−1.84	−1.8399543	−1.8396587	−1.840000009	8.99999986e−09
0.6	−1.64	−1.6400322	−1.640040	−1.64000036	3.60000000e−07
0.8	−1.36	−1.3599668	−1.35889879	−1.35999998	2.00000001e−08
1.0	−1.0	−0.9999476	−0.99987677	−0.99999999	1.00000001e−08

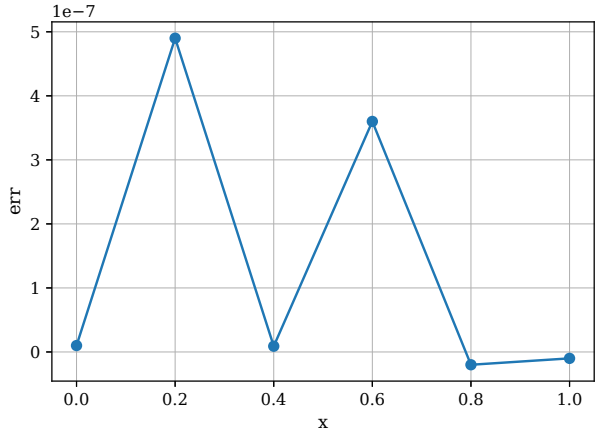
Uncorrected Proof



(A) Exact solution.



(B) Predicted solution.



(C) Absolute error.

FIGURE 4. Results of Experiment 4.3. Exact solution $y_t(x) = x^2 - 2$, predicted solution $y_p(x)$ by LDNN and absolute error $err = |y_t(x) - y_p(x)|$.

It has the exact solution $y(x) = x^2 + \frac{1}{2}$. The exact value, the predicted value and the absolute error (Error) in several test points in the $[0, 1]$ domain are reported in Table 5. 50 points of the shifted Legendre quadrature points are applied for training LDNN. The number of the training dataset is 500 and the number of the test dataset is 100. Figure 5 shows the illustrated comparison between y_t and y_p.

5. Theoretical convergence analysis of the LDNN approach

In this section, we provide a theoretical analysis of the convergence and error behavior of the proposed Legendre Deep Neural Network (LDNN) to solve the nonlinear Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs). The goal is to show that the LDNN approximation converges to the exact solution under mild smoothness and Lipschitz continuity assumptions.

5.1. **Preliminary Assumptions.**

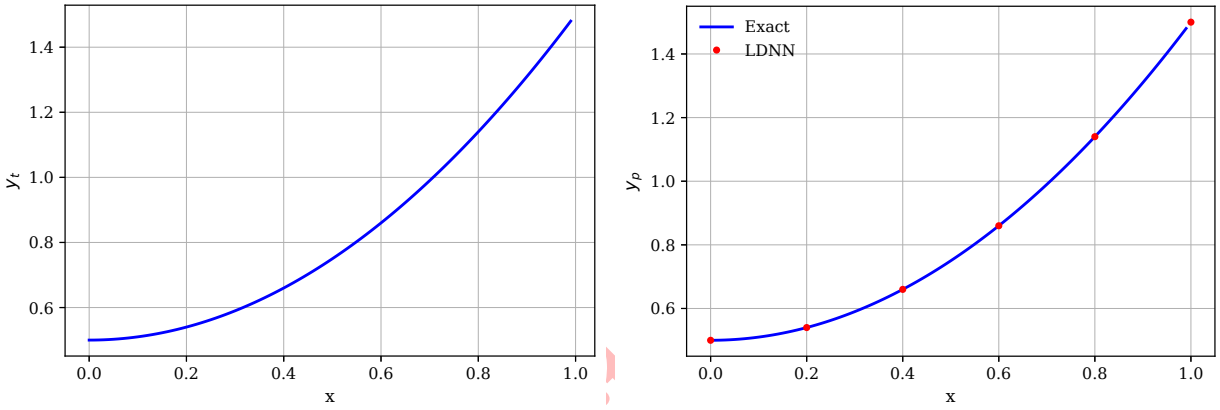
Let $y(x)$ denote the exact solution of the nonlinear integral equation

$$y(x) = g(x) + \xi_1 \int_0^x K_1(x, s)\phi_1(s, y(s))\, ds + \xi_2 \int_0^1 K_2(x, s)\phi_2(s, y(s))\, ds, \quad x \in [0, 1]. \tag{5.1}$$

We make the following assumptions:

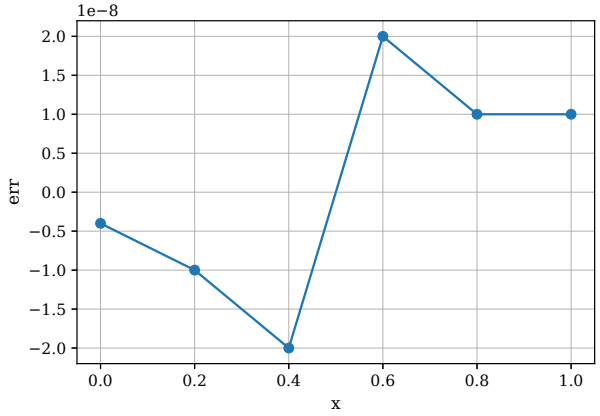
TABLE 5. The exact value y_t, the predicted value of the ADM method, the predicted value of the FNN, the predicted value y_p and the absolute error (Error $|y_t - y_p|$) of the LDNN in several test points on the $[0, 1]$ domain for Experiment 4.4.

x	exact value ($y_t = x^2 + \frac{1}{2}$)	predicted value by ADM	predicted value by FNN	predicted value (y_p) by LDNN	Error
0.0	0.5	0.50039285	0.50042379	0.50000004	4.00000000e−09
0.2	0.54	0.5400339	0.54006321	0.54000001	9.99999994e−09
0.4	0.66	0.6599865	0.6600465	0.66000002	2.00000000e−08
0.6	0.86	0.86001176	0.8598769	0.85999998	2.00000000e−08
0.8	1.14	1.1399317	1.13988365	1.13999999	9.99999994e−09
1.0	1.5	1.499708	1.4998377	1.49999999	9.99999994e−09



(A) Exact solution.

(B) Predicted solution.



(C) Absolute error.

FIGURE 5. Results of Experiment 4.4. Exact solution $y_t(x) = x^2 + \frac{1}{2}$, predicted solution $y_p(x)$ by LDNN and absolute error $err = |y_t(x) - y_p(x)|$.

(A1) $y(x) \in C^{p+1}[0,1]$ is sufficiently smooth.
(A2) $K_1(x,s), K_2(x,s) \in C^{p+1}([0,1]^2)$.
(A3) The nonlinear functions ϕ_1 and ϕ_2 are Lipschitz continuous, i.e.

$$|\phi_i(s,u) - \phi_i(s,v)| \le L_i|u-v|, \quad i = 1,2.$$

(A4) The Legendre polynomial basis of degree N provides an approximation error of order $O(N^{-q})$ for smooth functions, where q depends on the regularity of $y(x)$.

5.2. **Convergence Theorem.**

Theorem 5.1. *Let $y(x)$ be the exact solution of the V–F–H–IE and $y_{LDNN}(x)$ be its approximation obtained by the LDNN method with N Legendre neurons and a collocation grid of size h. Under assumptions (A1)–(A4), there exists a constant $C > 0$ independent of h and N such that*

$$\|y(x) - y_{LDNN}(x)\|_{L^2[0,1]} \le C\big(h^p + N^{-q} + \varepsilon_{train}\big), \tag{5.2}$$

where p is the order of the Legendre–Gauss quadrature, q denotes the spectral convergence rate, and ε_{train} represents the training (optimization) error of the network.

5.3. **Outline of the Proof.**

The total approximation error can be decomposed as

$$e(x) = y(x) - y_{LDNN}(x) = (y(x) - y_N(x)) + (y_N(x) - y_{LDNN}(x)), \tag{5.3}$$

where $y_N(x)$ is the projection of $y(x)$ onto the Legendre polynomial subspace. The first term corresponds to the spectral approximation error, which satisfies

$$\|y(x) - y_N(x)\|_{L^2[0,1]} = O(N^{-q}),$$

while the second term represents the neural approximation error, bounded by the training loss $\varepsilon_{\text{train}}$. Moreover, the collocation discretization introduces an additional error of order $O(h^p)$. Combining these results gives the stated bound.

5.4. **Discussion.**

The LDNN method thus achieves a hybrid convergence behavior:

- spectral accuracy with respect to the Legendre degree N,
- algebraic accuracy with respect to the collocation grid size h, and
- optimization-based refinement through $\varepsilon_{\text{train}}$.

Hence, the proposed LDNN framework inherits the strengths of both spectral collocation and deep learning, providing a highly accurate and efficient numerical solver for nonlinear V-F-H-IEs.

Uncorrected Proof

6. Conclusion

In this paper, a Legendre Deep Neural Network (LDNN) is introduced for solving nonlinear Volterra–Fredholm–Hammerstein integral equations (V-F-H-IEs). The LDNN consists of two coupled networks: the first is a feedforward neural network with an orthogonal hidden layer based on Legendre polynomial activation functions to approximate the solution $y(x)$, while the second network refines the output to satisfy the governing equation structure. Performance is enhanced using Legendre–Gaussian quadrature and automatic differentiation. Numerical experiments demonstrate the method's reliability, efficiency, and high accuracy. The LDNN method also exhibits hybrid convergence, combining spectral accuracy (via the Legendre degree N), algebraic accuracy (through grid size h), and further refinement from the training process. Consequently, it integrates the advantages of spectral collocation and deep learning, providing an accurate and efficient solver for nonlinear V-F-H-IEs.

References

[1] A. Afzal Aghaei, A. Ghasem Nejad, S. Yousefi, and K. Parand, *A new chebyshev operational matrix formulation of least-squares support vector regression for solving fractional integro-differential equations*, J. Comput. Appl. Math., *474* (2026), 116816.
[2] S. Amiri, M. Hajipour, and D. Baleanu, *A spectral collocation method with piecewise trigonometric basis functions for nonlinear volterra–fredholm integral equations*, Appl. Math. Comput., *370* (2020), 124915.

[3] A. Atangana, E. Bonyah, and A. A. Elsadany, *A fractional order optimal 4d chaotic financial model with mittag-leffler law*, Chin. J. Phys., (2020).
[4] E. Babolian, F. Fattahzadeh, and E. Golpar Raboky, *A chebyshev approximation for solving nonlinear integral equations of hammerstein type*, Appl. Math. Comput., *189*(1) (2007), 641–646.
[5] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, et al., *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*, Technical report, USDOE Office of Science (SC), Washington, DC (United States), (2019).
[6] H. Bassi, Y. Zhu, S. Liang, J. Yin, C. C. Reeves, V. Vlček, and C. Yang, *Learning nonlinear integral operators via recurrent neural networks and its application in solving integro-differential equations*, Mach. Learn. Appl., *15*(C) (2024), 100524.
[7] A.G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, J. Mach. Learn. Res., *18*(1) (2017), 5595–5637.
[8] S. Bazm and A. Hosseini, *Bernoulli operational matrix method for the numerical solution of nonlinear two-dimensional volterra–fredholm integral equations of hammerstein type*, Comput. Appl. Math., *39*(2)(2020), 1–20.
[9] H. Beiglo and M. Gachpazan, *Numerical solution of nonlinear mixed volterra-fredholm integral equations in complex plane via pqws*, Appl. Math. Comput., *369* (2020), 124828.
[10] J. Berg and K. Nyström, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, *317* (2018), 28–41.
[11] C. Canuto, M. Y. Hussaini, A. Quarteroni, and A. Thomas, Jr. *Spectral Methods in Fluid Dynamics*, Springer Science & Business Media, 2012.
[12] S. Chakraverty and S. Mall, *Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations*, CRC Press, 2017.
[13] R. Courant and D. Hilbert, *Methods of Mathematical Physics: Partial Differential Equations*, John Wiley & Sons, 2008.
[14] H. Dana Mazraeh, S. Eslami, A. Afzal Aghaei, and K. Parand, *A symbolic dataset for large language models to solve second-kind fredholm integral equations*, Sci. Data, *12*(1) (2025), 1930.
[15] W. Dang, Z. Gao, X. Sun, R. Li, Q. Cai, and C. Grebogi, *Multilayer brain network combined with deep convolutional neural network for detecting major depressive disorder*, Nonlinear Dyn., (2020), 1–11.
[16] H. T. Davis, *Introduction to Nonlinear Differential and Integral Equations*, US Government Printing Office, 1961.
[17] I. S. Gandzha and Y. V. Sedletsky, *A high-order nonlinear schrödinger equation as a variational problem for the averaged lagrangian of the nonlinear klein–gordon equation*, Nonlinear Dyn., *98*(1) (2019), 359–374.
[18] P. Görgel and A. Simsek, *Face recognition via deep stacked denoising sparse autoencoders (DSDSA)*, Appl. Math. Comput., *355* (2019),325–342.
[19] M. Hadizadeh and M. Mohamadsohi, *Numerical solvability of a class of volterra-hammerstein integral equations with noncompact kernels*, J. Appl. Math., *2005* (2005), 1–12.
[20] E. Hairer, C. Lubich, and M. Roche, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, Springer, 2006.
[21] Z. Hajimohammadi, F. Baharifard, and K. Parand, *A new numerical learning approach to solve general falkner–skan model*, Eng. Comput., (2020), 1–17.
[22] J. Han, A. Jentzen, and E. Weinan, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci. U.S.A., *115*(34) (2018), 8505–8510.
[23] B. Hanin, *Universal function approximation by deep neural nets with bounded width and ReLU activations*, Mathematics, *7*(10) (2019), 992.
[24] J. H. He and X. H. Wu, *Variational iteration method: New development and applications*, Comput. Math. Appl., *54*(7-8) (2007), 881–894.
[25] J. He, L. Li, J. Xu, and C. Zheng, *ReLU deep neural networks and linear finite elements*, arXiv preprint arXiv:1807.03973, (2018).
[26] M. H. Heydari, A. Atangana, and Z. Avazzadeh, *Chebyshev polynomials for the numerical solution of fractal–fractional model of nonlinear ginzburg–landau equation*, Eng. Comput., (2019), 1–12.

Uncorrected Proof

[27] Humaira, M. Sarwar, and T. Abdeljawad, *Existence of unique solution to nonlinear mixed volterra–fredholm–hammerstein integral equations in complex-valued fuzzy metric spaces*, J. Intell. Fuzzy Syst., *40*(3) (2021), 4065–4074.
[28] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Courier Corporation, 2012.
[29] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, ICLR, *1412* (2015).
[30] V. Krishnan, D. U. Sarwe, G. Pushpalakshmi, I. A. Mohammed, and J. Kishore, *Numerical methods for solving nonlinear fredholm integral equations in engineering applications*, In 2025 International Conference on Recent Innovation in Science Engineering and Technology (ICRISET), (2025).
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet classification with deep convolutional neural networks*, In Advanc. Neural Info. Proc. Sys.,(2012), 1097–1105.
[32] L. Kühnel, S. Sommer, and A. Arnaudon, *Differential geometry and stochastic dynamics with deep learning numerics*, Appl. Math. Comput., *365* (2019), 411–437.
[33] G. Lample and F. Charton, *Deep learning for symbolic mathematics*, arXiv preprint arXiv:1912.01412, (2019).
[34] L. Lardy, *A variation of nyström's method for hammerstein equations*, J. Integral Equations,(1981), 43–60.
[35] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, *521*(7553) (2015), 436–444.
[36] K. Li, J. Kou, and W. Zhang, *Deep neural network for unsteady aerodynamic and aeroelastic modeling across multiple mach numbers*, Nonlinear Dyn.,*96*(3) (2019), 2157–2177.
[37] S. Liao, *Homotopy Analysis Method in Nonlinear Differential Equations*, Springer, 2012.
[38] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., *45*(1-3) (1989), 503–528.
[39] Z. Long, Y. Lu, X. Ma, and B. Dong, *PDE-net: Learning PDEs from data*, ICML, (2018), 3208–3216.
[40] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, arXiv preprint arXiv:1907.04502, (2019).
[41] K. Maleknejad and E. Saeedipoor, *Convergence analysis of hybrid functions method for two-dimensional nonlinear volterra–fredholm integral equations*, J. Comput. Appl. Math., *368* (2020), 112533.
[42] S. Mall, S. K. Jeswal, and S. Chakraverty, *Connectionist learning models for application problems involving differential and integral equations*, Math. Methods Interdiscip. Sci., (2020), 1–22.
[43] H. R. Marzban, H. R. Tabrizidooz, and M. Razzaghi, *A composite collocation method for the nonlinear mixed volterra–fredholm–hammerstein integral equations*, Commun. Nonlinear Sci. Numer. Simul., *16*(3) (2011), 1186–1194.
[44] S. Mehrkanoon and J. A. Suykens, *Learning solutions to partial differential equations using LS-SVM*, Neurocomputing, *159* (2015), 105–116.
[45] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, *PPINN: Parareal physics-informed neural network for time-dependent PDEs*, Appl. Mech. Eng., *370* (2020), 113250.
[46] M. M. Moghaddam, H. Dana Mazraeh, and K. Parand, *A double deep Q network-powered approach to discovering symbolic solutions for nonlinear integral equations*, Eng. Appl. Artif. Intell., *169* (2026), 114113.
[47] H. Mohamadipanah, M. Heydari, and G. Chowdhary, *Deep kernel recursive least-squares algorithm*, Nonlinear Dyn., (2021), 1–16.
[48] M. Mohammad and C. Cattani, *A collocation method via the quasi-affine biorthogonal systems for solving weakly singular type of volterra-fredholm integral equations*, Alex. Eng. J., *59*(4) (2020), 2181–2191.
[49] A. Molina, P. Schramowski, and K. Kersting, *Padé activation units: End-to-end learning of flexible activation functions in deep networks*, ICLR, (2019).
[50] G. Pang, L. Lu, and G. E. Karniadakis, *fPINNs: Fractional physics-informed neural networks*, SIAM J. Sci. Comput., *41*(4) (2019), A2603–A2626.
[51] K. Parand and J. Amani Rad, *Numerical solution of nonlinear volterra–fredholm–hammerstein integral equations via collocation method based on radial basis functions*, Appl. Math. Comput., *218*(9) (2012), 5292–5309.
[52] K. Parand and M. Delkhosh, *Solving volterra's population growth model of arbitrary order using the generalized fractional order of the chebyshev functions*, Ric. Mat., *65*(1) (2016), 307–328.

Uncorrected Proof

[53] V. Peiris, N. Sharon, N. Sukhorukova, and J. Ugon, *Generalised rational approximation and its application to improve deep learning classifiers*, Appl. Math. Comput., *389* (2021), 125560.
[54] M. Pourbabaee, *Artificial neural networks method based on gegenbauer polynomials for solving some types of integral equations*, Comput. Appl. Math., *45*(3) (2026), 84.
[55] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys., *378* (2019), 686–707.
[56] M. A. Z. Raja, J. Mehmood, Z. Sabir, A. K. Nasab, and M. A. Manzar, *Numerical solution of doubly singular nonlinear systems using neural networks-based integrated intelligent computing*, Neural Comput. Appl., *31*(3) (2019), 793–812.
[57] M. Razzaghi and Y. Ordokhani, *A rationalized haar functions method for nonlinear fredholm–hammerstein integral equations*, Int. J. Comput. Math., *79*(3) (2002), 333–343.
[58] Z. Ren, S. Zhou, D. Liu, and Q. Liu, *EPINN: Enhanced physics-informed neural network for solving continuous integral equations*, J. Artif. Intell. Res., *84* (2025).
[59] J. Shen, T. Tang, and L. L. Wang, *Spectral methods: algorithms, analysis and applications, Vol. 41*, Springer Science & Business Media, 2011.
[60] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., *375* (2018), 1339–1364.
[61] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford University Press, 1985.
[62] S. Tang, B. Li, and H. Yu, *ChebNet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations*, arXiv preprint arXiv:1911.05467, (2019).
[63] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer science & business media, 2013.
[64] A. M. Wazwaz, *Linear and nonlinear integral equations, Vol. 639*, Springer, 2011.
[65] J. Xie, X. Gong, W. Shi, R. Li, W. Zhao, and T. Wang, *Applying the three-dimensional block-pulse functions to solve system of volterra–hammerstein integral equations*, Numer. Methods Partial Differ. Equ., *36*(6) (2020), 1648–1661.
[66] S. Yousefi and M. Razzaghi, *Legendre wavelets method for the nonlinear volterra–fredholm integral equations*, Math. Comput. Simul., *70*(1) (2005), 1–8.
[67] R. F. Zhang and S. Bilige, *Bilinear neural network method to obtain the exact analytical solutions of nonlinear partial differential equations and its application to p-gbkp equation*, Nonlinear Dyn., *95*(4) (2019), 3041–3048.

Uncorrected Proof